Унапређене верзије програма „Ка минималним паровима”

УДК 81’322.2

САЖЕТАК: Програм КаМП налази парове речи које су на сегменталном нивоу говора неподударне само по двама одабраним факторима (Deza and Deza 2016, 215) дужине бар по 1, нпр. *пе̑т* ∼ *пе̑ћ*, *фи̑лма* ∼ *фи̑рма*, *истȍризовати* ∼ *мајдȍризовати*, *пёсничкӣ* ∼ *полѝтичкӣ*. Предмет рада су брже варијанте КаМП-а са побољшаним сортирањем и са суплементарним модом.

КЉУЧНЕ РЕЧИ: фонетика, фонологија, обрада природног језика, корпусна лингвистика, Python.

РАД ПРИМЉЕН: 11. јануар 2022.
РАД ПРИХВАЋЕН: 11. мај 2022.

Данило Алексић
danilo.aleksic@fil.bg.ac.rs
Универзитет у Београду
Филолошки факултет
Београд, Србија

Лазар Мркела
lazar.mrkela@metropolitan.ac.rs
Универзитет Метрополитан
Факултет информационих технологија
Београд, Србија

1. Увод

Према (Bugarski 2003, 128), минимални парови су парови „kod kojih se dve po značenju odelite reči formalno razlikuju samo u jednoj fonemi“, нпр. *бȁс* ∼ *чȁс*.[1] Програм Ка минималним паровима (Алексић and Шандрих 2021) у српском корпусу налази парове речи узајамно формално различитих само по задатим подниска̄ма[2], при чему занемарује прозодију и то да ли су слова велика или мала. Корпус треба да буде кодиран схемом UTF-8. Осим задатих подниски, у „речима“ могу

1. Ивић (1961–1962, 75) помиње прозодијске системе „у којима се јавља на хиљаде минималних парова речи одн. *облика диференцираних искључиво прозодијским контрастима* [курзив – Д. А.]“. Такви парови су *вȉла* ∼ *вúла*, *лȏза* (генитив од *лоз* ‘срећка’) ∼ *лдза* и др.

2. Уопште важи да је свака ниска подниска саме себе (Partee, Meulen, and Wall 1993, 433; Singh 2009, 33) и да дужина ниске може бити 1 (Partee, Meulen, and Wall 1993, 432; Python 2021b). Дакле, подниском не би било погрешно називати (I) ниску `"ima"` из регуларног израза `\b\S*ima\S*\b` посматрану у односу на упарену ниску `"ima"`, ни (II) карактер `"a"`.

бити (I) карактери од `"A"` до `"Z"`, од `"a"` до `"z"` и од `"Ć"` до `"ž"` у одговарајућим табелама Unicode-а и (II) цртице у медијалном положају.

Садржај улазне датотеке	**Задате подниске**	**Ниска за излаз**
`"Klima-uređaji pre klima-uređaja"`	`"a", "i"`	`"klima-uređaja` ∼ `Klima-uređaji"` (или `"Klima-uređaji` ∼ `klima-uređaja"`)
`"α-čestica, α-čestice, α-čestici"`	`"a", "e"`	`"čestica` ∼ `čestice"`
`"α-čestica, β-čestica"`	`"α", "β"`	`"α-čestica` ∼ `β-čestica"`

Табела 1. КаМП: примери улаза и излаза

Презентовани програм може бити од користи професорима српског као страног језика и лингвистима (Алексић and Шандрих 2021, 574–75).

Област	**Задате подниске**	**Утилизација**
Настава српског као страног језика	`"c", "č"`	Темељ питања у вежби: „*Ц* или *ч*? 1) Шта би ти урадио, драги читао_е? Наше новине ће поштовати своје читао_е. 2) [...]“
Дериватологија	`"auto", "samo"`	Подаци о конкуренцији сегмената *аутo-* и *само-*.

Табела 2. КаМП: примери утилизације

Овом приликом аутори објављују и коментаришу унапређене верзије КаМП-а које су изградили, КаМП 2 и КаМП 2.1.[3]

3. В. Додатак 1.

КаМП 2 и КаМП 2.1 алати су из домена обраде природног језика ако се она схвати „у широком смислу“, на тај начин да подразумева „било коју врсту рачунарске манипулације природним језиком“ (Bird, Klein, and Loper 2009, ix). Наиме, дотични програми не акцентују ни мали број свих српских речи, него српски језик процесирају површно. Будући да су донекле подешени за претрагу обимних корпуса, нови КаМП-ови су скромни прилози и корпусној лингвистици ако се она дефинише нпр. као „рачунарски потпомогнута анализа врло опсежних збирки транскрибованих говорних јединица или писаних текстова“ (McEnery and Hardie 2012, i).

КаМП 2 и КаМП 2.1 кодирани су у Python-у 3.8.2 (Python 2021a). Python је интерпретирани виши програмски језик опште намене (Pajankar 2020, 52). Овај језик је „и елегантан и прагматичан, и једноставан и моћан“; „погодан је за почетнике у програмирању и одличан за стручњаке“ (Martelli, Ravenscroft, and Holden 2017, ix). Python постаје све популарнији, и у 2017. години постао је најпопуларнији језик на свету према магазину IEEE Spectrum (Shovic and Simpson 2021, 1). Python је најшире употребљавани програмски језик у обради природног језика (Antić 2021, vii). Може се очекивати да Python буде спор у поређењу са компајлираним језицима, али је он бржи ако се у обзир не узме само колико траје извршење кода него и колико траје развијање кода (Unpingco 2021, 2). Python је у касним 80-им годинама прошлог века створио холандски програмер Гвидо ван Росум (Cicolani 2021, 41; Rajagopalan 2021, 1).

2. Сродни ресурси

У (Алексић and Шандрих 2021, 569) наведена су четири алата за налажење минималних парова одн. „фонолошких суседа“ (Mairano and Calabrò 2016, 258) која су направљена пре КаМП-а. Њима се могу прибројати програмски пакет за Python 3 Minpair (PyPI 2021) и кратки програм на Python-у 2.7 са странице (Stack Overflow 2021a).

Minpair по најмање двама задатим „вокалским фонолошким елементима“ тражи „минималне парове (и минималне скупове) [само једносложних — Д. А.] речи из америчког енглеског“, (А) `defaultdict`-ом групишући те речи према прикљученим транскрипцијама у којима је (Б) задате вокале, помоћу регуларног израза и функције `enumerate()`, претходно заменио тачком.

Приступ А има начелну паралелу у КаМП-у 2.1, али (према Додатку 2) нешто ефикаснију. КаМП 2.1 речи упарује помоћу стандардног речника (в. Додатак 1).

Приступ Б има начелну паралелу у КаМП-у 2 и КаМП-у 2.1, али (према Додатку 3) много ефикаснију. КаМП 2 и КаМП 2.1 задате елементе специјалном ниском замењују помоћу метода `str.replace()` и `str.format()` (в. Додатак 1).

Унос из `cmudict-a`	**Торка за груписање ако су задати вокали `"AE"` и `"OW"`**
`("cat", ["K", "AE1", "T"])` `("coat", ["K", "OW1", "T"])`	`("K", ".", "T")`

Табела 3. Minpair: пример улаза и торке за груписање

```
# Minpair: примери употребе 1 и 2
import minpair
print(minpair.vowel_minpair(["AO", "ER"])[12:13])
# Излаз: [{'AO': 'saw', 'ER': 'sir'}]

print(minpair.vowel_minpair(["AA", "AO", "EH"])[6:7])
# Излаз: [{'AO': 'dawn', 'EH': 'den', 'AA': 'don'}]
```

Могу се задати и врста речи или врсте речи.

```
# Minpair: примери употребе 3, 4 и 5
import minpair
print(minpair.generator(pos=["ADV"]).vowel_minpair(
        ["AH", "EH"]))
# Излаз: [{'AH': 'once', 'EH': 'whence'}]

print(minpair.generator(
    pos=["ADJ", "VERB"]).vowel_minpair(
    ["AE", "IH"])[:1])
# Излаз: [{'AE': 'bad', 'IH': 'bid'}]

print(minpair.generator(
    pos=["ADJ", "VERB"]).vowel_minpair(
    ["AE", "IH"])[22:23])
# Излаз: [{'AE': 'sang', 'IH': 'sing'}]
```

Извор(е) речи није могуће задати. Minpair „зависи од неколиких корпуса са платформе NLTK“: „*brown*, *cmudict*, *universal_ tagset* и *words*“.

Кодом понуђеним на страници (Stack Overflow 2021a) упарују се ниске које се разликују по једном карактеру, а исте су дужине. Ниске морају бити унутар нпр. листе, али се пред њих не постављају никакви природнојезички услови (није нужно да оне буду из одређеног језика, односно на одређеном писму, па ни да буду сачињене од алфабетских карактера). Током извршења кода, свака задата ниска пореди се карактер по карактер са сваком задатом ниском иза ње. Ако се поклапају сви карактери осим једног, дотични пар ниски се исписује.

```
# (Stack Overflow 2021a)
for n1,word1 in enumerate(wordlist):
    for word2 in wordlist[n1+1:]:
        if len(word1)==len(word2):
            ndiff=0
            for n,letter in enumerate(word1):
                if word2[n]!=letter:
                    ndiff+=1
            if ndiff==1:
                print word1, word2

"""Програм са странице (Stack Overflow 2021a): пример употребе 1
(додао Д. А.)
Улаз: ["kula", "kule", "kuli", "kulom"]
Излаз:
kula kule
kula kuli
kule kuli
"""
```

Ниједан карактер осим оног што уноси разлику не може у првом члану излазног пара бити мало, а у другом велико слово и обрнуто.

```
"""Програм са странице (Stack Overflow 2021a): примери употребе 2 и 3
Улаз: ["kula", "kulE", "kuli", "kulom"]
Излаз:
kula kulE
kula kuli
kulE kuli

Улаз: ["kula", "Kule", "kuli", "kulom"]
Излаз:
```

```
kula kuli
"""
```

Када је у улазној листи прва ниска била дуплирана, у излазу се јавио дуплиран пар.

```
"""Програм са странице (Stack Overflow 2021a): пример употребе 4
Улаз: ["kula", "kula", "kule", "kulom"]
Излаз:
kula kule
kula kule
"""
```

	Minpair	**(Stack Overflow 2021a)**	**КаМП 2 и КаМП 2.1**
Излаз су парови ниски различитих по **било ком** карактеру на датој позицији.	×	✓	×
Диференцијални елементи се задају.	✓	×	✓
Број задатих диференцијалних елемената не мора бити 2.	✓		×
Задати диференцијални елементи не морају бити вокали.	×		✓
Речи не мора диференцирати низ од само једне фонеме одн. од само једног карактера.	×	×	✓
Улаз бира корисник.	×	✓	✓
Улаз не мора бити већ токенизован.		×	✓

Табела 4. КаМП 2 / КаМП 2.1 спрам сродних алата

3. Важније новине у КаМП-у 2 и КаМП-у 2.1

У **припремном делу алгоритма** формирају се торке за поредбу речи, нпр. (`"Avali", "avali", "▯v▯li"`), (`"Požeškom", "požeškom", "pož▯škom"`), (`"sekretarijata", "sekretarijata", "s▯kr▯t▯rij▯t▯"`). Први члан торке за поредбу јесте реч која садржи једну или обе задате подниске. Други члан је први члан пребачен у мала слова. Трећи члан је други члан у ком је бар једна потврда прве или друге задате подниске замењена ниском "▯"[4]. Једној ексцерпираној речи могу одговарати једна реч са заменом задатих подниски (в. Табелу 5) или, (I) када се задате подниске преклапају (Lothaire 2005, 7) или (II) када је једна задата подниска права подниска (Böckenhauer and Bongartz 2007, 24) друге задате подниске, више речи са заменом задатих подниски (в. Табелу 6).

		Ниске за поредбу
Ексцерпирана реч	**Ексцерпирана реч пребачена у мала слова**	**Ексцерпирана реч пребачена у мала слова уз замењивање задатих подниски**
`"Knjiga"`	`"knjiga"`	`"knjig▯"`
`"knjigu"`	`"knjigu"`	`"knjig▯"`
`"sveska"`	`"sveska"`	`"svesk▯"`
`"SVESKU"`	`"svesku"`	`"svesk▯"`
`"računaljku"`	`"računaljku"`	`"r▯č▯n▯ljk▯"`

Табела 5. КаМП 2 / КаМП 2.1: примери ниски за поредбу (ако су задате подниске `"a"` и `"u"`)

У **главном делу алгоритма** формиране торке се пореде. КаМП 2 генерише све могуће двочлане комбинације торки са првом задатом подниском и торки са другом задатом подниском и онда прескаче нежељене комбинације. Могуће двочлане комбинације овај програм добија тако што израчунава Декартов производ[5] поменутих двеју група

4. Она је одабрана јер је упадљива и релативно неуобичајена. Те су особине, наравно, присутне и код многих других ниски.

5. На пример, Декартов производ скупа ниски `{"број", "улица"}` и скупа ниски `{"МЕСТО", "ОПШТИНА"}` јесте скуп уређених парова

	Ниске за поредбу	
Ексцерпирана реч	**Ексцерпирана реч пребачена у мала слова**	**Ексцерпирана реч пребачена у мала слова уз замењивање једне задате подниске**
`"ONA"`	`"ona"`	`"on▀"`
`"Onima"`	`"onima"`	`"on▀"`
`"Onima"`	`"onima"`	`"onim▀"`
`"onimima"`	`"onimima"`	`"onim▀"`
`"onimima"`	`"onimima"`	`"onimim▀"`

Табела 6. КаМП 2 / КаМП 2.1: примери ниски за поредбу (ако су задате подниске `"a"` и `"ima"`)

торки. КаМП 2.1 од торки са другом задатом подниском образује хеш-мапу (табелу) и проверава да ли се у њој сређу речи са заменом узете из торки са првом задатом подниском. Кључ мапе је реч са заменом, док је вредност мапе мапа речи од којих се добија тај исти кључ. – КаМП 2 и КаМП 2.1 исписују оне парове ексцерпираних речи чији се чланови (I) разликују када се пребаце у мала слова и (II) подударају по речима са заменом, дакле парове речи које се разликују само по задатим поднискама. На пример, ако су задате подниске `"a"` и `"u"`, а у улазној датотеци је само ниска `"Knjiga, knjigu, sveska, SVESKU"`, КаМП 2 и КаМП 2.1 неће исписати ниске `"Knjiga ∼ SVESKU"` и `"knjigu ∼ sveska"`, пошто ниска `"knjig▀"` није једнака нисци `"svesk▀"`, него ће исписати ниске `"Knjiga ∼ knjigu"` и `"sveska ∼ SVESKU"`.

КаМП 2 и КаМП 2.1 имају (А) мод у ком игноришу разлике између великих и малих слова али фаворизују ниске малих слова и (Б) мод у ком би нпр. ексцерпиране ниске `"vitraž"` и `"Vitraž"` обрадили као засебне речи (в. Табелу 7). Разлог је следећи оправдани коментар из (Алексић and Шандрих 2021, 574): „Поставља се само питање важности величине слова.“ На пример, у настави српског као страног језика властите именице некад имају приоритет над невластитим речима. Име *Чак* (*Бери*, *Норис*...) погодно је за вежбу изговора са фотографијама; каква би фотографија дочарала значење непроменљиве

ниски `{("број", "МЕСТО"), ("број", "ОПШТИНА"), ("улица", "МЕСТО"), ("улица", "ОПШТИНА")}`.

речи *чак*? У моду Б, КаМП 2 и КаМП 2.1 из корпуса `POL` исписују не само пар `"čak ∼ Žak"` него и пар `"Čak ∼ Žak"` (уз `"Čak ∼ ŽAK"` итд.).[6]

Садржај улазне датотеке	`"EUPRAVE, eUprave, euprave, EUPRAVA, eUprava, euprava"`
Ниска за излаз КаМП-а	`"EUPRAVA ∼ EUPRAVE"` (или `"EUPRAVE ∼ EUPRAVA"`)
Ниска за излаз КаМП-а 2 и КаМП-а 2.1 у моду А	`"euprava ∼ euprave"`
Ниске за излаз КаМП-а 2 и КаМП-а 2.1 у моду Б	`"euprava ∼ euprave"`, `"euprava ∼ eUprave"`, `"euprava ∼ EUPRAVE"`, `"eUprava ∼ euprave"`, `"eUprava ∼ eUprave"`, `"eUprava ∼ EUPRAVE"`, `"EUPRAVA ∼ euprave"`, `"EUPRAVA ∼ eUprave"`, `"EUPRAVA ∼ EUPRAVE"`

Табела 7. КаМП и КаМП 2 / КаМП 2.1: (не)разликовање великих и малих слова (ако су задате подниске `"a"` и `"e"`)

Функција `segmentacija_korpusa()`[7] реорганизована је. Она корпус више не учитава помоћу бесконачне `while`-петље,[8] него, по угледу на пример препорученог начина за позивање функције до стражарске вредности из (Hettinger 2021, 12.27 и даље), помоћу `for`-петље, која је „брза и лепа“.

6. Велико почетно слово не гарантује да `"Čak"` у `POL`-у може бити име (због реченица као `"Čak sam pronašao i kupca."`). Доказе да може пружа накнадна претрага (`"Čak Blekvel"`, `"Čak Dejli"`, `"Čak Noris"`...).

7. Посреди је генераторска функција која, да би се штедела радна меморија, улазни корпус сегментира „тако да се не цепају речи“ (Алексић and Шандрих 2021, 572, 581).

8. Уп. кôд унет 14. маја 2021. у одговор од 11. јуна 2015. са странице (Stack Overflow 2021b).

КаМП нађене парове сортира према Unicode-кодним позицијама простих слова, док КаМП 2 и КаМП 2.1 нађене парове сортирају по позицијама простих слова у нискама `mali_alfabet` и `veliki_alfabet` (в. Табелу 8).

```
# Ниске за сортирање у КаМП-у 2 и КаМП-у 2.1
mali_alfabet = "- ~abcčćdđefghijklmnopqrsštuvwxyzž"
veliki_alfabet = "- ~ABCČĆDĐEFGHIJKLMNOPQRSŠTUVWXYZŽ"
```

Улазна листа	["nota ∼ note", "đaka ∼ đake", "Bač ∼ Beč"]
Улазна листа сортирана онако како КаМП сортира парове	["Bač ∼ Beč", "nota ∼ note", "đaka ∼ đake"]
Улазна листа сортирана онако како КаМП 2 и КаМП 2.1 сортирају парове	["Bač ∼ Beč", "đaka ∼ đake", "nota ∼ note"]

Табела 8. КаМП и КаМП 2 / КаМП 2.1: сортирање парова

Истина, и нови КаМП-ови сортирајући парове користе Unicode-кодне позиције, али само код карактера којих нема у нискама за сортирање (в. Табелу 9).

Пар	**Листа за сортирање**	**Порекло броја**
α	945	Unicode
-	0	Ниска
z	32	mali_alfabet
r	23	
a	3	
č	6	
e	10	
n	19	
j	15	
e	10	
	1	
∼	2	
	1	
β	946	Unicode
-	0	Ниска
z	32	mali_alfabet
r	23	
a	3	
č	6	
e	10	
n	19	
j	15	
e	10	

Табела 9. КаМП 2 / КаМП 2.1: пример листе за сортирање

КаМП 2 и КаМП 2.1 сортирају чланове сваког пара пре него што их споје у ниску за излаз (нпр. `["knjigu", "Knjiga"]` → `["Knjiga", "knjigu"]`).

4. Брзина извршавања

Брзина је мерена у Python-у 3.8.2, на Manjaro Linux-у, рачунаром са процесором i5-11600K и два DDR4-3200 CL16 SDRAM-а од по 16 GB и на корпусу `POL`, који „броји око 117.900.900 речи из 223.308 текстова са сајта *Политика*“ (Алексић and Шандрих 2021, 575).

Задате подниске	Брзина у секундама (просек пет сукцесивних мерења)				
	КаМП	КаМП 2		КаМП 2.1	
		Мод А	Мод Б	Мод А	Мод Б
"č", "ć"	334	199	246	67	66
"dž", "đ"	135	81	85		
"nadnad", "supersuper"	6639	**65,79**	**65,51**	66,38	66,09
"ir", "zir"	143	86	90	**67** (!)	66
"ma", "va"	2153	1257	1516		

Табела 10. КаМП, КаМП 2 и КаМП 2.1: брзина извршавања

5. Кратка оцена ефикасности КаМП-а 2[9]

Функција која проналази парове у КаМП-у 2 заснована је на Декартовом производу две листе. Овај начин представља елегантно решење у смислу прегледности и комплексности кода, али због квадратног понашања није довољно ефикасан у случају листи са великим бројем елемената. Проблем се лако уочава из експерименталних резултата, где се примећује значајно дуже време извршавања у случају подниски које се чешће појављују (ma-va) (уп. Табелу 10).

9. Одељке 5 и 6 написао је Л. Мркела. Остале одељке (без оних двеју реченица из Одељка 3 што се тичу искључиво КаМП-а 2.1), Додатак 2 и Додатак 3 написао је Д. Алексић.

6. Даљи рад

У реалним условима, који могу захтевати да се овај програм покреће на слабијим рачунарима, само учитавање корпуса и издвајање речи које садрже тражене подниске може да траје превише дуго. На пример, учитавање корпуса POL.xml на једном старијем лаптоп рачунару (Acer Aspire 3, Intel Quad Core N3710, 4GB RAM) траје и до приближно 15 минута. Предлог је да се дода и опција креирања речника од корпуса који би се сачувао на диску. Ова обрада корпуса би се извршила само једном, а касније би се речник користио за проналажење парова за нове подниске.

Следећи могући корак у скраћивању времена извршавања јесте паралелизација претраге. Данас и слабији рачунари имају више „језгара“ у оквиру процесора (на пример, рачунар из претходног пасуса има 4 језгра). Зато је могуће неке делове кода паралелно извршавати и тиме додатно убрзати програм. Један предлог за једноставну паралелизацију јесте подела једне од листи речи на n делова, а затим обрада тих делова на различитим процесорима (језгрима) паралелно. Како је и секвенцијална верзија са хеширањем већ веома ефикасна, прво треба решити проблем са спорим учитавањем корпуса, па тек онда размишљати о даљим убрзањима.

7. Закључак

У поређењу са КаМП-ом, КаМП 2 и КаМП 2.1 за краће време постижу више — налазе практично исте парове и нађене парове и речи унутар нађених парова још сортирају по бољем методу.

КаМП 2.1 је у већини испитаних случајева био неспорно бржи од КаМП-а 2.

Додатак 1. КаМП 2 и КаМП 2.1[10]

```
"""КаМП 2.1 је модификована верзија КаМП-а 2. КаМП 2 је
пак модификована верзија КаМП-а.
Упаривање у функцијама КаМР_2_1_a() и КаМР_2_1_b()
допринос је Л. Мркеле, а остатак кода (са функцијама
ekscerp(), obrada_reči_1() и obrada_reči_2()) допринос је
Д. Алексића.
"""


def main():
    from functools import partial
    from itertools import product
    import re
    import sys

    sys.stdout.reconfigure(encoding="utf-8")
    """B. (Алексић and Шандрих 2021, 580)."""
    prvo_slovo = "ma".casefold()
    drugo_slovo = "va".casefold()
    preklapanje = False
    razl_vel_i_mal_slova = False
    sort_znak = chr(1114111)
    mali_alfabet = "- abcčćdđefghijklmnopqrsštuvwxyzž"
    veliki_alfabet = "- ABCČĆDĐEFGHIJKLMNOPQRSŠTUVWXYZŽ"

    def spajanje_niski(*niske):
        return "".join(niske)

    def segmentacija_korpusa(
            korpus, veličina=8192, separator="\n"):
        """Уп. (581).
        Функција враћа делове корпуса задате
        величине по задатом сепаратору.
        """
        ostatak = ""
        for komad in iter(
```

10. Главни извор била је званична документација, са сајта (Python 2021a). Други извори су индицирани упућивањем на линкове из Литературе, упућивањем на одговарајућа места у (Алексић and Шандрих 2021) и упућивањем на једно место у овом раду.

```
                partial(korpus.read, veličina), ""):
            """B. (Hettinger 2021, 12.27 и даље)."""
            komad = spajanje_niski(ostatak, komad)
            if separator in komad:
                delovi = komad.rsplit(separator, 1)
                """B. (W3Schools 2021)."""
                yield delovi[0]
                ostatak = delovi[1]
            else:
                ostatak = komad
        if ostatak:
            yield ostatak

    def prvo_mala_slova_1(reč):
        """Кључ за сортирање речи који фаворизује речи
        сачињене од малих слова.
        """
        if reč.islower():
            return "!"
        elif reč.istitle():
            return sort_znak
        else:
            for slovo in reč:
                if slovo.isupper():
                    reč = reč.replace(slovo, sort_znak)
            return reč

    def prvo_mala_slova_2(reč):
        """Кључ за сортирање нађених парова. На врху
        листе ће бити парови који садрже мање великих слова.
        """
        if reč.islower():
            return "!"
        else:
            reč = reč.replace(" ~ ", "")
            if reč.istitle():
                return sort_znak
            else:
                for slovo in reč:
                    if slovo.isupper():
                        reč = reč.replace(
                            slovo, sort_znak)
                return reč
```

```
80
81      def indeksiranje_za_listu(reč):
82          """Уп. (Stack Overflow 2021c).
83          Кључ за сортирање по задатом редоследу.
84          """
85          lista_za_sort = []
86          for slovo in reč:
87              if slovo in mali_alfabet:
88                  lista_za_sort.append(
89                      mali_alfabet.index(slovo))
90              elif slovo in veliki_alfabet:
91                  lista_za_sort.append(
92                      veliki_alfabet.index(slovo))
93              else:
94                  lista_za_sort.append(ord(slovo))
95          return lista_za_sort
96
97      def prosta_zamena_slova(reč):
98          """У речима се задате подниске замењују
99          специјалном ниском.
100         """
101         if prvo_slovo in reč and drugo_slovo not in reč:
102             reč_sa_zamenom = reč.replace(
103                 prvo_slovo, "\u23B2")
104         elif prvo_slovo not in reč and drugo_slovo in reč:
105             reč_sa_zamenom = reč.replace(
106                 drugo_slovo, "\u23B2")
107         elif prvo_slovo in reč and drugo_slovo in reč:
108             reč_sa_zamenom = reč.replace(
109                 prvo_slovo, "\u23B2")
110             reč_sa_zamenom = reč_sa_zamenom.replace(
111                 drugo_slovo, "\u23B2")
112         return (reč_sa_zamenom,)
113
114     def složena_zamena_slova(reč, slovo):
115         """У речима се задате подниске замењују
116         специјалном ниском.
117         Покривају се случајеви када између
118         задатих подниски има преклапања.
119         В. (Алексић and Шандрих 2021, 580--81).
120         """
121         izlazni_skup = set()
122         reč_za_obradu = reč.replace(slovo, "{}")
```

```
123         for kombinacija in product(
124                 [slovo, "\u23B2"],
125                 repeat=reč_za_obradu.count("{}")):
126             reč_sa_zamenama = reč_za_obradu.format(
127                 *kombinacija)
128             if reč_sa_zamenama != reč:
129                 izlazni_skup.add(
130                     reč_sa_zamenama)
131         return izlazni_skup
132 
133     def zamena_slova(reč):
134         if not preklapanje:
135             return prosta_zamena_slova(reč)
136         else:
137             skup = set()
138             skup.update(
139                 složena_zamena_slova(reč, prvo_slovo),
140                 složena_zamena_slova(reč, drugo_slovo))
141             return skup
142 
143     def tokenizacija():
144         """Корпус се претвара у речник речи издвојених помоћу
145         регуларног израза. Избегнута је употреба врло сложених
146         регуларних израза у случајевима када су задате
147         подниске дуже (в. Табелу 10).
148         Уп. Одељак 6.
149         """
150         rečnik = {}
151         with open(r"/.../POL.xml",
152                   "r", encoding="utf-8") as korpus:
153             komadi = segmentacija_korpusa(korpus)
154             for komad in komadi:
155                 pogoci = re.findall(
156                     "[A-Za-zĆ-ž-\u00ad]+", komad)
157                 """B. (573--74)."""
158                 for pogodak in pogoci:
159                     reč = pogodak.strip("-")
160                     if "\u00ad" in reč:
161                         reč = reč.replace("\u00ad", "")
162                         """B. (581)."""
163                     rečnik[reč] = reč.casefold()
164         return rečnik
165 
```

```
166     def ekscerp(slovo):
167         """Из речника добијеног од корпуса узимају се речи
168         које садрже задату подниску.
169         """
170         return (ključ
171                 for ključ, vrednost
172                 in rečnik_od_korpusa.items()
173                 if slovo in vrednost)
174 
175     def dekart(lista_1, lista_2):
176         """Елиминишу се нежељени парови из Декартовог
177         производа обрађених речи.
178         """
179         return (
180             (*sorted([b, e]), a, d)
181             for (a, b, c), (d, e, f)
182             in filter(
183             lambda torka: torka[0][2] == torka[1][2]
184                           and torka[0][1] != torka[1][1],
185             product(lista_1, lista_2, repeat=1)))
186 
187     def obrada_reči_1(gen):
188         """Враћају се торке у којима су речи, речи пребачене
189         у мала слова и речи са заменама.
190         Ова функција се позива када се жели занемарити
191         разлика између великог и малог слова.
192         """
193         lista_torki = []
194         brojač = set()
195         lista_reči = sorted(
196             list(gen), key=prvo_mala_slova_1)
197         for reč in lista_reči:
198             reč_malim_slovima = rečnik_od_korpusa[reč]
199             if reč_malim_slovima not in brojač:
200                 brojač.add(reč_malim_slovima)
201                 for reč_sa_zamenom in zamena_slova(
202                         reč_malim_slovima):
203                     lista_torki.append(
204                         (reč, reč_malim_slovima,
205                          reč_sa_zamenom))
206         return lista_torki
207 
208     def obrada_reči_2(gen):
```

```
209         """Враћају се торке у којима су речи, речи пребачене
210         у мала слова и речи са заменама.
211         Ова функција се позива када се НЕ ЖЕЛИ занемарити
212         разлика између великог и малог слова.
213         """
214         lista_torki = []
215         for reč in gen:
216             reč_malim_slovima = rečnik_od_korpusa[reč]
217             for reč_sa_zamenom in zamena_slova(
218                     reč_malim_slovima):
219                 lista_torki.append(
220                     (reč, reč_malim_slovima,
221                      reč_sa_zamenom))
222         return lista_torki
223 
224     def KaMP_2_a():
225         """Завршна обрада у КаМП-у 2 ако се жели занемарити
226         разлика између великог и малог слова.
227         """
228         brojač = set()
229         for torka in dekart(
230                 obrada_reči_1(ekscerp(
231                     prvo_slovo)),
232                 obrada_reči_1(ekscerp(
233                     drugo_slovo))):
234             if (torka[0], torka[1]) not in brojač:
235                 brojač.add((torka[0], torka[1]))
236                 konačni_skup.add((torka[2], torka[3]))
237         lista_parova = [
238             " ~ ".join(sorted(list(torka),
239                               key=indeksiranje_za_listu))
240             for torka in konačni_skup]
241         lista_parova.sort(key=indeksiranje_za_listu)
242         for par in lista_parova:
243             print(par)
244         print("\n\tBROJ PAROVA:")
245         print("\t\t", len(lista_parova))
246 
247     def KaMP_2_b():
248         """Завршна обрада у КаМП-у 2 ако се НЕ ЖЕЛИ занемарити
249         разлика између великог и малог слова.
250         """
251         konačni_skup = {(torka[2], torka[3])
```

```
                        for torka in dekart(
                obrada_reči_2(ekscerp(
                    prvo_slovo)),
                obrada_reči_2(ekscerp(
                    drugo_slovo)))}
        lista_parova = [
            " ~ ".join(sorted(list(torka),
                              key=indeksiranje_za_listu))
            for torka in konačni_skup]
        lista_parova = list(set(lista_parova))
        lista_parova.sort(key=prvo_mala_slova_2)
        lista_parova.sort(key=indeksiranje_za_listu)
        for par in lista_parova:
            print(par)
        print("\n\tBROJ PAROVA:")
        print("\t\t", len(lista_parova))

    def KaMP_2_1_a():
        """Упаривање речи и завршна обрада у КаМП-у 2.1
        ако се жели занемарити разлика између великог
        и малог слова.
        """
        lista1 = obrada_reči_1(ekscerp(
            prvo_slovo))
        lista2 = obrada_reči_1(ekscerp(
            drugo_slovo))
        mapa = {}
        for x in lista2:
            if x[2] not in mapa:
                mapa[x[2]] = {}
            mapa[x[2]][x[0]] = x[1]
        for torka in lista1:
            result = mapa.get(torka[2])
            if result is not None:
                for k, v in result.items():
                    if (torka[1] != v and (k, torka[0])
                            not in konačni_skup):
                        konačni_skup.add((torka[0], k))
        lista = []
        for par in konačni_skup:
            par = list(par)
            par.sort(key=indeksiranje_za_listu)
            izlaz = spajanje_niski(par[0], " ~ ", par[1])
```

```
295             lista.append(izlaz)
296         lista.sort(key=indeksiranje_za_listu)
297         for par in lista:
298             print(par)
299         print("Broj parova: ", len(lista))
300 
301     def KaMP_2_1_b():
302         """Упаривање речи и завршна обрада у КаМП-у 2.1
303         ако се НЕ ЖЕЛИ занемарити разлика између великог
304         и малог слова.
305         """
306         lista1 = obrada_reči_2(ekscerp(
307             prvo_slovo))
308         lista2 = obrada_reči_2(ekscerp(
309             drugo_slovo))
310         mapa = {}
311         for x in lista2:
312             if x[2] not in mapa:
313                 mapa[x[2]] = {}
314             mapa[x[2]][x[0]] = x[1]
315         for torka in lista1:
316             result = mapa.get(torka[2])
317             if result is not None:
318                 for k, v in result.items():
319                     if (torka[1] != v and (k, torka[0])
320                             not in konačni_skup):
321                         konačni_skup.add((torka[0], k))
322         lista = []
323         for par in konačni_skup:
324             par = list(par)
325             par.sort(key=indeksiranje_za_listu)
326             izlaz = spajanje_niski(par[0], " ~ ", par[1])
327             lista.append(izlaz)
328         lista.sort(key=prvo_mala_slova_2)
329         lista.sort(key=indeksiranje_za_listu)
330         for par in lista:
331             print(par)
332         print("Broj parova: ", len(lista))
333 
334     if (prvo_slovo[-1:] == drugo_slovo[:1]
335             or prvo_slovo[:1] == drugo_slovo[-1:]
336             or (prvo_slovo in drugo_slovo
337                 or drugo_slovo in prvo_slovo)):
```

```
        preklapanje = True
    konačni_skup = set()
    rečnik_od_korpusa = tokenizacija()
    if razl_vel_i_mal_slova:
        KaMP_2_1_b()  # КаМП 2 позива функцију KaMP_2_b().
    else:
        KaMP_2_1_a()  # КаМП 2 позива функцију KaMP_2_a().


if __name__ == "__main__":
    main()
```

Додатак 2. Minpair vs. КаМП 2.1: брзина упаривања

```
"""Приступ из Minpair-а.
"""
from collections import defaultdict

mapa_1 = defaultdict(lambda: {})
for x in lista_2:
    mapa_1[x[2]][x[0]] = x[1]

"""Приступ из КаМП-а 2.1.
"""
mapa_2 = {}
for x in lista_2:
    if x[2] not in mapa_2:
        mapa_2[x[2]] = {}
    mapa_2[x[2]][x[0]] = x[1]

"""Улаз је била листа торки типа ("subsidiaries",
"subsidiaries", "subsidi.ri.s"), направљена од
речи са "a" и/или "e" из cmudict-а.

Приступ из КаМП-а 2.1 показао се око 7% бржим
у Python-у 3.8.2 на систему описаном у Одељку 4.
Поређени су проседи 500 сукцесивних мерења
(55 ms : 51 ms).
"""
```

Додатак 3. Minpair vs. КаМП 2 / КаМП 2.1: брзина замењивања

```
"""Приступ из Minpair-а.
"""
vowels_regex = re.compile(r'^(?:%s)' % '|'.join(vowels))
matches = [vowels_regex.search(phone) for phone in word]
list_with_repl = []
for i, character in enumerate(word):
    for j, match in enumerate(matches):
        if i == j:
            if match:
                list_with_repl.append(".")
            else:
                list_with_repl.append(character)
string_with_repl = "".join(list_with_repl)

"""Приступ из КаМП-а 2 и КаМП-а 2.1.
"""
реч_sa_zamenom = reč.replace(
    prvo_slovo, ".")
reč_sa_zamenom = reč_sa_zamenom.replace(
    drugo_slovo, ".")

"""Улаз су биле речи са "a" и/или "e" из
cmudict-а.

Приступ из КаМП-а 2 и КаМП-а 2.1 показао се
око 95% бржим у Python-у 3.8.2 на систему
описаном у Одељку 4. Поређени су просеци
500 сукцесивних мерења (472 ms : 23 ms).
Међутим, мора се истаћи да у Minpair-у кôд
за замењивање добија листу, а враћа торку
(нпр. ["L", "UW", "S"] → ("L", ".", "S")),
док је у КаМП-у 2 и КаМП-у 2.1 и улаз и излаз
кода за замењивање -- ниска
(нпр. "teorijska" → "t◸orijsk◸").
"""
```

Литература

Antić, Zhenya. 2021. *Python Natural Language Processing Cookbook: Over 50 recipes to understand, analyze, and generate text for implementing language processing tasks.* Birmingham: Packt Publishing.

Bird, Steven, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python.* Sebastopol, CA: O'Reilly Media.

Böckenhauer, Hans-Joachim, and Dirk Bongartz. 2007. *Algorithmic Aspects of Bioinformatics.* Berlin: Springer.

Bugarski, Ranko. 2003. *Uvod u opštu lingvistiku.* 2nd ed. Beograd: Čigoja štampa.

Cicolani, Jeff. 2021. *Beginning Robotics with Raspberry Pi and Arduino: Using Python and OpenCV.* 2nd ed. Berkeley, CA: Apress.

Deza, Michel Marie, and Elena Deza. 2016. *Encyclopedia of Distances.* 4th ed. Berlin: Springer.

Hettinger, Raymond. 2021. "Transforming Code into Beautiful, Idiomatic Python." Accessed August 21, 2021. https://www.youtube.com/watch?v=OSGv2VnC0go.

Lothaire, M. 2005. *Applied Combinatorics on Words.* Cambridge: Cambridge University Press.

Mairano, Paolo, and Lidia Calabrò. 2016. "Are minimal pairs too few to be used in pronunciation classes?" In *La fonetica nell'apprendimento delle lingue: Phonetics and language learning,* edited by Renata Savy and Iolanda Alfano, 255–268. Milano: Officinaventuno.

Martelli, Alex, Anna Ravenscroft, and Steve Holden. 2017. *Python in a Nutshell.* 3rd ed. Sebastopol, CA: O'Reilly Media.

McEnery, Tony, and Andrew Hardie. 2012. *Corpus Linguistics: Method, Theory and Practice.* Cambridge: Cambridge University Press.

Pajankar, Ashwin. 2020. *Raspberry Pi Computer Vision Programming: Design and implement computer vision applications with Raspberry Pi, OpenCV, and Python 3.* 2nd ed. Birmingham: Packt Publishing.

Partee, Barbara H., Alice ter Meulen, and Robert E. Wall. 1993. *Mathematical Methods in Linguistics.* Dordrecht: Kluwer Academic Publishers.

PyPI. 2021. "minpair 0.1.3." Accessed October 26, 2021. https://pypi.org/project/minpair.

Python. 2021a. Accessed August 21, 2021. https://www.python.org.

Python. 2021b. "Text Sequence Type — `str`." Accessed August 21, 2021. https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str.

Rajagopalan, Gayathri. 2021. *A Python Data Analyst's Toolkit: Learn Python and Python-based Libraries with Applications in Data Analysis and Statistics.* Berkeley, CA: Apress.

Shovic, John C., and Alan Simpson. 2021. *Python All-in-One For Dummies.* 2nd ed. Hoboken, NJ: John Wiley & Sons.

Singh, Arindama. 2009. *Elements of Computation Theory.* London: Springer.

Stack Overflow. 2021a. "Finding (phonological) minimal pairs with python." Accessed August 31, 2021. https://stackoverflow.com/q/26157361.

Stack Overflow. 2021b. "Lazy Method for Reading Big File in Python?" Accessed August 31, 2021. https://stackoverflow.com/q/519633.

Stack Overflow. 2021c. "Sorting string values according to a custom alphabet in Python." Accessed October 26, 2021. https://stackoverflow.com/q/26579392.

Unpingco, José. 2021. *Python Programming for Data Analysis.* Cham: Springer.

W3Schools. 2021. "Python String rsplit() Method." Accessed October 26, 2021. https://www.w3schools.com/python/ref_string_rsplit.asp.

Алексић, Данило, and Бранислава Шандрих. 2021. "Аутоматска ексцерпција парова речи за учење изговора у настави српског као страног језика." *Српски језик: студије српске и словенске* 26 (1): 567–584. ISSN: 0354-9259. https://doi.org/10.18485/sj.2021.26.1.32. http://doi.fil.bg.ac.rs/pdf/journals/sj/2021-1/sj-2021-26-1-32.pdf.

Ивић, Павле. 1961–1962. "Број прозодијских могућности у речи као карактеристика фонолошких система словенских језика." *Јужнословенски филолог* 25: 75–113.